The Gap Between BDI Agents and Semantic Hypermedia and What We Can Do About It

Samuele Burattini[1,*,†], Martina Baiardi[1,†], Giovanni Ciatto[1] and Danilo Pianini[1]

[1]Department of Computer Science and Engineering, Alma Mater Studiorum—Università di Bologna

Abstract

Traditional BDI agents, rooted in logic programming, remain challenging to integrate with the hypermedia nature of open Web environments that rely on Semantic Web technologies like RDF and OWL. This paper examines the gap between these paradigms and surveys existing integration efforts on a conceptual and technical level and argues that existing tools offer limited ergonomic support for developers. Our proposal for a deeper integration relies on a generalized BDI engine to enable the development of BDI agents that can directly reason and operate on semantic hypermedia resources. We derive ideal requirements and show an abstract architecture that can be tailored to support different types of beliefs and reasoning mechanisms.

Keywords

BDI, Hypermedia, Semantic Web, Agent-Oriented Programming

1. Introduction

Integrating intelligent agents with the Web is a long-standing goal in both the multi-agent system (MAS) [1] and Semantic Web [2] communities, but, despite such common interest, the famous question "Where are all the intelligent agents?" [3], remains relevant today. Although McBurney and Luck responded by arguing that agents were not a "killer application" and that "once dynamic and open systems become the norm" companies would adopt agent technologies as a fundamental [4]. More than fifteen years later the Web has transitioned towards a more dynamic and open environment, but it is still dominated by traditional clients, servers, and (most recently) bots and scrapers—which can hardly be considered full agents. However, with the widespread adoption of the REpresentational State Transfer (REST) architectural style [5] and Web interoperability standards like Web of Things (WoT) [6], the integration of MAS and the Web has gained renewed momentum.

Along this line, the World Wide Web Consortium (W3C) Autonomous Agents on the Web community group[1] has been founded to connect researchers and practitioners and a new perspective has been proposed: hypermedia multi-agent system (hMAS) [7] envision the (Semantic) Web as an open environment in which agents can interact by consuming and manipulating hypermedia resources.

Even more recently, with the advent of large language model (LLM)-based *Agentic AI* [8] the creation of LLM-driven agents that interact with Web APIs [9] (or directly with Websites [10]) has gained significant attention. However, despite being promising, these approaches do not make structured knowledge representation obsolete [11], and often trade the controllability offered by traditional agent-programming paradigms for more guarantees in terms of correctness, and completeness.

Arguably, Web-integrated agents developed with cognitive architectures, such as belief-desire-intention (BDI) agents [12], would retain controllability and predictability: they could directly exploit hypermedia to discover new resources and actions, and use Semantic Web technologies like Resource

The Second International Workshop on Hypermedia Multi-Agent Systems (HyperAgents 2025), in conjunction with the 28th European Conference on Artificial Intelligence (ECAI 2025); October 26, 2025, Bologna, Italy

*Corresponding author.

†These authors contributed equally.

samuele.burattini@unibo.it (S. Burattini); m.baiardi@unibo.it (M. Baiardi); giovanni.ciatto@unibo.it (G. Ciatto); danilo.pianini@unibo.it (D. Pianini)

0009-0009-4853-7783 (S. Burattini); 0009-0001-0799-9166 (M. Baiardi); 0000-0002-1841-8996 (G. Ciatto); 0000-0002-8392-5409 (D. Pianini)

[1]https://www.w3.org/community/webagents/

Description Framework (RDF) [13] and Web Ontology Language (OWL) [14] to act intelligently without compromising on quality. However, we believe the integration of BDI agents with the Web is hindered by a conceptual and technological gap, which limits developers in creating agents that need to interact with (Semantic) hypermedia. More precisely: **(G1)** BDI agents are historically tied to Logic Programming (LP) whereas the Semantic Web is grounded on Description Logic (DL), and although the two paradigms are theoretically compatible, their practical implementations interoperate poorly, increasing integration cost significantly; **(G2)** BDI agents are typically based on the procedural reasoning system (PRS) architecture [15] which relies on pre-defined plans and does not natively support discovering new actions at runtime, a necessary feature to interact proficiently with hypermedia environments.

In this paper, we reflect on these gaps, their technical implications, explore how this gap is currently being addressed in the hMAS community, and propose a mitigation strategy based on abstracting the design of agent-oriented programming languages and frameworks.

2. Background

2.1. BDI Agents

The BDI model [12] is a well-established cognitive architecture for the development of intelligent agents. BDI agents follow a continuous cycle of *(i) perception* of external stimuli and gather new *beliefs*, *(ii) deliberation* to update beliefs, goals and choose *intentions* to pursue selecting *plan* to achieve goals, and *(iii) action* advancing one intention by executing a step of the associated plan. Practically, most BDI agents implementations follow the PRS architecture [15], which is based on the ability for agents to select plans from a predefined set and execute them to achieve their goals. The focus on BDI agents, as opposed to other reasoning-loop cognitive architectures, is motivated by their direct mapping between high-level mental attitudes (beliefs, desires, intentions) and practical agent programming constructs (plans and actions). This makes BDI agents particularly suitable for engineering controllable, predictable, and explainable agent behaviors. These properties, might be desirable in Web environments, where agents need to reliably interact with humans and other systems.

To declaratively define the agent's behavior the agent-oriented programming (AOP) community has developed several BDI programming languages and frameworks [16]. Arguably, the most popular language is AgentSpeak(L) [17] with its Jason [18] implementation, which uses a formal syntax and semantics derived from LP for representing beliefs, goals, and actions as logic predicates.

Although BDI programming languages and frameworks with little or no reliance on LP mechanisms exist (e.g., [19, 20, 21]), they come with their own semantics and representation for beliefs, goals, and actions which still leave an abstraction gap to be covered with respect to Semantic Web technologies. In the remainder of this work we will refer to BDI agents implicitly assuming AgentSpeak(L)-like agents.

2.2. The Web, Semantics and Hypermedia

The World Wide Web was proposed by Tim Berners-Lee as a global, open information space based on the ability to link resources through hypermedia documents [22]. The Web is built with the REST architectural style [5] which supports independently evolving components.

The Semantic Web [23] complements the *self-descriptive* principle of REST by encoding semantics in the representation through the general model of RDF [13] based on machine-readable *triples* that state facts using a common vocabulary defined within ontologies. OWL [14] allows the encoding of such ontologies following the principles of DL to define concepts and relationships between them.

Another fundamental principle is Hypermedia as the Engine of Application State (HATEOAS) which defines hypermedia as "the simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions" [5]. This way, the Web can be seen as an open environment where clients discover new resources and available actions by consuming hypermedia. When such hypermedia is annotated with explicit semantics, it

should be possible for clients to better understand the meaning of the affordances, avoiding the need for hard-coded knowledge.

2.3. Hypermedia Multi-Agent Systems

The idea of hMAS [7] seeks a deeper integration of MAS and the Web through the adoption of hypermedia and REST principles. The main idea behind such integration is to overcome the limitations of previous integration approaches by considering the Web as an *environment* for agents to operate in [24]. Considering the environment as a first-class abstraction [25], hMAS revisits the Agents and Artifacts meta-model [26] as a way to modularize and program the environment. In hMAS, artifacts are represented as hypermedia resources, exposing affordances that agents can discover through the uniform interface of the Web. Most efforts in the hMAS community have focused on shaping the environment dimension, since, ideally, heterogeneous agents should be able to operate within it. However, the agent dimension also deservers further investigation to improve how agents can effectively exploit the hypermedia environment (e.g., for action discovery at runtime [27]). Thus, in this paper, we focus on the agent dimension of hMAS, and specifically on BDI agents and how to support deeper integration with semantic hypermedia.

3. Integrating BDI Agents and Semantic Hypermedia: a Gap Analysis

Here we delve into the details of the gaps that we identified between BDI agents and semantic hypermedia, namely, **(G1)** the practical differences for knowledge representation and manipulation between LP and Semantic Web technologies, and **(G2)** the limitations of BDI agents with *open* hypermedia environments, where actions are not predefined but discovered at runtime.

Syntactical Differences Between LP and RDF. On a syntactical level, BDI agents commonly represent beliefs as definite clauses [28] (e.g. `person(alice)`). In the Semantic Web, knowledge is represented in the form of RDF *triples* which connect a *subject* to an *object* through a *predicate* (e.g. `:alice rdf:type foaf:Person`). Although RDF triples can be represented as definite clauses [29] (while the opposite is not true), in practice, semantic knowledge is available on the Web in RDF, and accessing that from BDI agents requires a translation step, which has many degrees of freedom (e.g., `type(alice, person)` and `triple(alice, type, person)` are reasonable mappings of the RDF triple above), which is completely up to BDI developers. In other words, implementing the translation is additional work which, if handled manually in an ad-hoc manner can be source of inconsistencies.

Ontological Inference. RDF triplets are commonly paired with OWL ontologies (e.g., `foaf`[2] in the example above), providing *axioms* that define the meaning of the triples's predicates (e.g., `rdf:type` denotes the *instance of* relation between subjects and classes), as well as constraints, and relationships between concepts. This is a way to standardize knowledge representation across hypermedia. Logic programs, instead, are not really meant to be scattered over the Web, nor being accessed remotely.

Again, BDI developers who want to create agents capable of ontological inference face a hidden challenge: either they translate the OWL ontology into definite clauses, or they wrap some OWL reasoner into the agent interpreter and reasoning cycle. Despite being automatable to some extent (cf. [30]), both activities require additional engineering effort, and should rather be addressed with direct support from the BDI framework.

Querying. BDI commonly relies on Selective Linear Definite clause (SLD) resolution [31] to query the belief base, when this is implemented as a logic program. Furthermore, they rely on logic unification [32] to match plans and goals, making the intertwining with LP even deeper. Conversely, the Semantic Web relies on SPARQL Protocol and RDF Query Language (SPARQL) [33] to query RDF data. SPARQL

[2] http://xmlns.com/foaf/spec/

allows matching graph patterns and retrieve all the values of variables in the pattern from the matched triples, providing an intuitive way to query the knowledge base.

Despite both LP and SPARQL allow expressing declarative queries for a knowledge base, the latter may be more practical and efficient for RDF data, especially when the query spans over multiple triples. Consider for instance a query searching for the unknown relations `?r` between `alice` and `bob`, and matching all the triples where the subject is `carl` and the predicate is `?r` to retrieve all the objects `?x`. A query of this kind would be straightforward to write in SPARQL, while it would require second-order variables in LP. This feature is not commonly supported by LP or BDI frameworks, and is typically worked around via *meta-programming*, resulting in longer and more cumbersome LP code.

As for inference, given the expressiveness limitations of OWL, Semantic Web Rule Language (SWRL) [34] can be used to define inference rules. Here, a similar argument like the one above concerning the wrapping of SWRL reasoners applies.

Open vs. Close Environments. BDI agents are equipped with a predefined set of plans, composed of sequences of actions that the agent knows how to execute. Additionally, BDI agents usually run in a closed environment, which is designed by the developer to provide perceptions to the agents and support the execution of the actions used in the plans [25]. The development process of BDI agents is hence based on the fact that a developer anticipates the relevant possible settings an agent may encounter and provides plans to handle them. This makes it difficult for agents to adapt to unexpected settings or to flexibly incorporate new actions that become available while exploring the environment. Relating this to the open hypermedia nature of the Web, makes the effective integration of BDI agents challenging. Following the HATEOAS principle, agents may discover new affordances (i.e., possible actions) while navigating through the hypermedia environment. The inability to exploit such new affordances may limit the agent in adapting to the environment and effectively interact with it.

3.1. Integration Requirements

Guided by our gap analysis, we identify a set of requirements that we believe may improve practical development of BDI agents in Web environments. We hence propose that a BDI agent programming framework for hMAS should support the following features: **(R1)** direct manipulation of RDF and OWL triples in the agent's belief base; **(R2)** ontological inference for deliberation (e.g., plan selection and execution); **(R3)** support for querying the belief base via SPARQL; **(R4)** ability to exploit affordances discovered in the environment to dynamically adapt the agents' plans.

4. Related Works

Here we summarize some related works demonstrating how, despite the many extensions of AgentSpeak towards DL, the direct exploitation of Semantic Web technologies (namely, **(R1)** and **(R3)**) in BDI *programming* frameworks is still underexplored.

In [35] authors propose AgentSpeak-DL, an extension of AgentSpeak to support the $\mathcal{ALC}$ description logic [36]. There, agents' belief bases consist of an $\mathcal{ALC}$ ontology (concepts, roles, and individuals) and any operation involving access to the belief base (e.g., plan selection, belief update, and query) imply either performing inference or updating such ontology. So, AgentSpeak-DL provides the theoretical foundations to support our integration requirements from Section 3.1. AgentSpeak-DL is then extended by CooL-AgentSpeak [37], where authors explore inter-agent exchange of procedural knowledge [38], possibly expressed by different ontologies. The authors of [39] provide a different theoretical integration of AgentSpeak with DL to account for *goal states* (new, suspended, active, succeeded, failed) in the agent's lifecycle, in such a way that state transitions are tied to conditions expressed in DL.

AgentSpeak-DL also serves as the foundation for JASDL [40], an implementation of AgentSpeak-DL in Jason [18]. The key extension in JASDL is the introduction of "semantically-enriched literals," i.e., $\mathcal{ALC}$ statements expressed as logic facts, (see example in Listing 1). JASDL adopts Jason belief syntax and

Listing 1: On the left, RDF triples expressed as predicates in JASDL, on the right their meaning inferred by a OWL reasoner. The example is taken from [40].

```
hotel(hilton)[o(travel)].                          // hilton is a hotel
hasRating(hilton, threeStarRating)[o(travel)].     // hilton has three-star rating
city(london)[o(travel)].                           // london is a city
```

```
// plain RDF form
targetEquipment(Machine) :-
  rdf("https://territoire.emse.fr/kg/emse/fayol", "https://w3id.org/bot#containsElement", Machine) &
  rdf(Machine, "http://www.w3.org/1999/02/22-rdf-syntax-ns#type", "http://www.productontology.org/id/Filler_(
      packaging)") .

// equivalent short form based on OntologyArtifact properties
targetEquipment(Machine) :- containsElement(fayol, Machine) & 'Filler_(packaging)'(Machine).
```

Listing 2: The example shows how RDF triples are represented in Hypermedea agents belief base. The example is taken from Hypermedea Application Programming Interface (API) reference (https://archive.is/vxULu).

belief-base management mechanisms, while wrapping and adapting a semantic reasoner to let agents perform ontological inference, thus addressing **(R2)**. With this approach, developers gain ontological inference in Jason, but at the price of translating RDF triples into logic predicates.

Other approaches [37, 41, 42, 43] rely on encapsulating functionalities and services in external components that agents can share and exploit to support their activities. For instance CooL-AgentSpeak [37] relies on CArtAgO [44], to define and implement the *Ontology Artifact*, providing agents with ontological inference capabilities—encoding DL statements as beliefs similarly to JASDL.

Similarly, Hypermedea [41] extends JaCaMo [45] (again, via artifacts and constraints on the beliefs-base syntax) to support agents interacting with semantic hypermedia and, in particular, the WoT. More precisely, Hypermedea *artifacts* produce agent beliefs such as the ones in Listing 2 through observable properties—i.e., RDF triples expressed as logic predicates, in Jason's syntax. Furthermore, other *artifacts* support the manipulation of such RDF triples, materialization of ontological inferences, as well as the interaction with WoT's *things*, and the synthesis of plans via a PDDL planner—hence addressing **(R4)**.

A different take is proposed in [42] where Astra [46] agents are equipped with a set of *modules* to allow BDI agents to store knowledge and interact with a "personal" triple store separated from the belief base. Similarly to other approaches, the bridge to belief representation is by mapping triples directly to logic predicates with three arguments.

Finally, although not directly related to BDI agents, the Yggdrasil framework [24] provides a hypermedia layer for building hMAS environments on top of CArtAgO artifacts and has been used to implement hMAS applications with the JaCaMo framework.

Exploiting Discovered Actions at Runtime. To exploit new actions discovered at runtime **(R4)** BDI agents must have access to a *planner* either within the agent (see [47] for a general overview of planning in BDI agents) or through the environment—as proposed in Hypermedea [41]. In the context of hMAS, this requires that the environment formally describes how and when to use them (e.g., their preconditions and effects), and that agents can interpret such information to instruct the planner accordingly. Recent works explored this idea by using *signifiers* [48] in hypermedia environments to better convey affordances to BDI agents [27]. A less structured approach is proposed in [49, 50], where agents (albeit not BDI) are able to select actions in a hypermedia environment by using a LLM oracle.

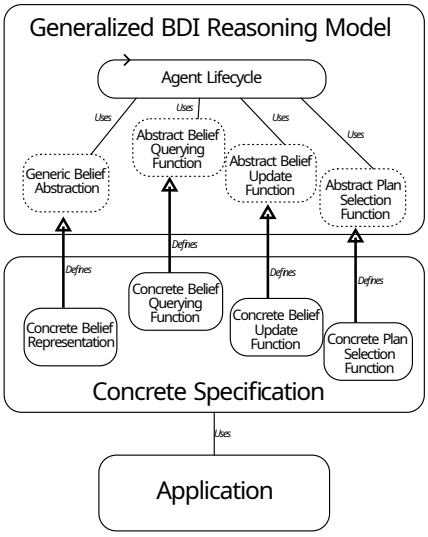
5. Levelling the Gap with a Generalized BDI Engine

To bridge the gap between BDI agents and semantic hypermedia, we propose a *generalized* BDI engine, addressing the requirements in Section 3.1. This approach primarily tackles gap **(G1)**, while leaving

the exploration of gap **(G2)** (benefitting from the modularity of the engine) to future works. A similar approach has been proposed in [51], in which authors discuss the idea of a modular BDI architecture, to make it independent of the logic representation and reasoning mechanisms, although not in the context of hypermedia. Sharing the core modularity idea, we propose to create a BDI interpreter encapsulating agents reasoning cycle, while decoupling the specific technology used for knowledge representation and manipulation. This would allow for tailoring the reasoning mechanisms to application-specific needs. In this way, beliefs can flexibly support different formats depending on the domain at hand and be adapted to work directly with e.g., RDF triples, to JSON, YAML, and other Web standards. Such a design fulfills the first requirement **(R1)**.

However, separating the reasoning cycle of BDI agents from knowledge representation is complex, as the two are deeply intertwined [35]. Consider for instance these fundamental operations: *(i) plan selection* depending on conditions to be tested against the belief base; *(ii) belief querying* as part of a plan, for the sake of decision-making; *(iii) belief update* for the sake of updating the agent's knowledge, as part of a course of action. In LP-based frameworks, these would simply rely on logic unification and resolution, whereas in a generalized BDI engine, each operation may rely on different matching strategies. For instance, for Semantic Hypermedia agent, one may rely on RDF and OWL for beliefs representation, SPARQL for belief querying and update, and SWRL for plan selection and ontological inference—thus addressing **(R2)** and **(R3)**.

Figure 1: Generalized Architecture.



The generalized BDI engine can be realized with a layered architecture (see Figure 1) comprising three layers. *Layer 1*: **generalized BDI reasoning layer**, where the agent's deliberation process is implemented as a reasoning cycle, yet agnostic to how beliefs are represented and manipulated. This is where agents' intentions are maintained and updated as agents perceive, decide what to do, and finally act. Any operation involving belief bases should rely on some *abstract* API, to be implemented by the next layer. *Layer 2*: **concrete specification layer**, where the aforementioned API is implemented to provide concrete operations on the belief base. This layer acts as an adapter between the generalized BDI reasoner and the specific knowledge representation technologies of choice. Lastly, *Layer 3*: **application layer**, where actual goals, plans, and interactions of a MAS are defined to tackle some specific use case.

Requirement **(R4)** although not directly addressed by the approach, could be integrated with the techniques already being explored in the hMAS community (cf. Section 4), benefitting from the possibility of choosing the most suitable knowledge representation technology to facilitate their agents to discover new actions at runtime in the hypermedia environment.

6. Conclusion

In this paper, we present a gap analysis between BDI agents and semantic hypermedia, showing how they are difficult to integrate due to the different technologies and paradigms they rely on. We identify a set of ideal requirements, finding that none of the surveyed related works fully satisfy them and propose to implement them through a generalized BDI engine that can be tailored to use RDF and OWL as knowledge representation technologies.

Future work will focus on implementing the architecture proposed in Section 5 and explore how to support hMAS developers with syntax extensions and tools to ease the development of BDI agents in hypermedia environments within our BDI programming framework JaKtA [52].

Acknowledgments

This work has been partially supported by: *(i)* "WOOD4.0 - Woodworking Machines for Industry 4.0", Emilia-Romagna CUP E69J22007520009; *(ii)* "FAIR–Future Artificial Intelligence Research", Spoke 8 "Pervasive AI" (PNRR, M4C2, Investimento 1.3, Partenariato Esteso PE00000013), funded by the EC under the NextGenerationEU programme; *(iii)* "ENGINES — ENGineering INtElligent Systems around intelligent agent technologies" project funded by the Italian MUR program "PRIN 2022" (G.A. 20229ZXBZM), and *(iv)* 2023 PhD scholarship (PNRR M4C2, Investimento 3.3 DM 352/2022), co-funded by the European Commission and AUSL della Romagna.

Declaration on Generative AI

During the preparation of this work, the authors used GitHub's Copilot in order to spell-check their writing, and make it more concise. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] M. O. Shafiq, Y. Ding, D. Fensel, Bridging multi agent systems and web services: towards interoperability between software agents and semantic web services, in: Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society, 2006, pp. 85–96. doi:10.1109/EDOC.2006.18.

[2] O. Lassila, J. Hendler, T. Berners-Lee, The semantic web, Scientific american 284 (2001) 34–43.

[3] J. A. Hendler, Where are all the intelligent agents?, IEEE Intell. Syst. 22 (2007) 2–3. doi:10.1109/MIS.2007.62.

[4] P. McBurney, M. Luck, The agents are all busy doing stuff!, IEEE Intell. Syst. 22 (2007) 6–7. doi:10.1109/MIS.2007.77.

[5] R. T. Fielding, R. N. Taylor, Principled design of the modern web architecture, ACM Trans. Internet Techn. 2 (2002) 115–150. doi:10.1145/514183.514185.

[6] M. Lagally, et al., Web of Things (WoT) Architecture, W3C Recommendation, World Wide Web Consortium, 2023. URL: https://www.w3.org/TR/2023/REC-wot-architecture11-20231205/.

[7] A. Ciortea, et al., A decade in hindsight: The missing bridge between multi-agent systems and the World Wide Web, in: E. Elkind, et al. (Eds.), Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), IFAAMAS, 2019, pp. 1659–1663.

[8] D. B. Acharya, K. Kuppan, D. Bhaskaracharya, Agentic AI: autonomous intelligence for complex goals - A comprehensive survey, IEEE Access 13 (2025) 18912–18936. doi:10.1109/ACCESS.2025.3532853.

[9] Y. Du, F. Wei, H. Zhang, AnyTool: Self-reflective, hierarchical agents for large-scale API calls, in: Proceedings of the 41st International Conference on Machine Learning, ICML'24, JMLR.org, 2024.

[10] B. Zheng, B. Gou, J. Kil, H. Sun, Y. Su, GPT-4V(ision) is a generalist web agent, if grounded, in: Proceedings of the 41st International Conference on Machine Learning, ICML'24, JMLR.org, 2024.

[11] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Trans. Knowl. Data Eng. 36 (2024) 3580–3599. doi:10.1109/TKDE.2024.3352100.

[12] A. S. Rao, M. P. Georgeff, Modeling rational agents within a BDI-architecture, in: J. F. Allen, R. Fikes, E. Sandewall (Eds.), Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann, 1991, pp. 473–484.

[13] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014, W3C Recommendation, World Wide Web Consortium, 2014. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[14] B. Motik, P. F. Patel-Schneider, B. Parsia, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012, W3C Recommendation, World Wide Web Consortium, 2012. URL: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[15] M. P. Georgeff, A. L. Lansky, Procedural knowledge, Proc. IEEE 74 (1986) 1383–1398. doi:10.1109/PROC.1986.13639.

[16] V. Mascardi, D. Demergasso, D. Ancona, Languages for programming BDI-style agents: an overview, in: F. Corradini, F. D. Paoli, E. Merelli, A. Omicini (Eds.), WOA 2005 - Workshop "From Objects to Agents", Pitagora Editrice Bologna, 2005, pp. 9–15. URL: http://lia.deis.unibo.it/books/woa2005/papers/2.pdf.

[17] A. S. Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, in: W. V. de Velde, J. W. Perram (Eds.), Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, volume 1038 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 42–55. doi:10.1007/BFB0031845.

[18] R. H. Bordini, J. F. Hübner, M. Wooldridge, Programming multi-agent systems in AgentSpeak using Jason, John Wiley & Sons, 2007.

[19] A. Pokahr, L. Braubach, W. Lamersdorf, Jadex: A BDI reasoning engine, in: R. H. Bordini, M. Dastani, J. Dix, A. E. F. Seghrouchni (Eds.), Multi-Agent Programming: Languages, Platforms and Applications, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Springer, 2005, pp. 149–174.

[20] T. Kampik, J. C. Nieves, JS-son - A lean, extensible JavaScript agent programming library, in: L. A. Dennis, R. H. Bordini, Y. Lespérance (Eds.), Engineering Multi-Agent Systems (EMAS) 7th International Workshop, volume 12058 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 215–234. doi:10.1007/978-3-030-51417-4_11.

[21] N. Howden, R. Rönnquist, A. Hodgson, A. Lucas, JACK intelligent agents-summary of an agent infrastructure, in: 5th International conference on autonomous agents, volume 6, 2001, p. 40.

[22] T. Berners-Lee, R. Cailliau, J. Groff, The World-Wide Web, Comput. Networks ISDN Syst. 25 (1992) 454–459. doi:10.1016/0169-7552(92)90039-S.

[23] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American 284 (2001) 34–43. URL: http://dx.doi.org/10.1038/scientificamerican0501-34. doi:10.1038/scientificamerican0501-34.

[24] A. Ciortea, O. Boissier, A. Ricci, Engineering World-Wide multi-agent systems with hypermedia, in: D. Weyns, V. Mascardi, A. Ricci (Eds.), Engineering Multi-Agent Systems (EMAS) 6th International Workshop, volume 11375 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 285–301. doi:10.1007/978-3-030-25693-7_15.

[25] D. Weyns, A. Omicini, J. Odell, Environment as a first class abstraction in multiagent systems, Auton. Agents Multi Agent Syst. 14 (2007) 5–30. doi:10.1007/S10458-006-0012-0.

[26] A. Ricci, M. Piunti, M. Viroli, Environment programming in multi-agent systems: an artifact-based perspective, Auton. Agents Multi Agent Syst. 23 (2011) 158–192. doi:10.1007/S10458-010-9140-7.

[27] D. Vachtsevanou, et al., Enabling BDI agents to reason on a dynamic action repertoire in hypermedia environments, in: M. Dastani, et al. (Eds.), Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM, 2024, pp. 1856–1864. doi:10.5555/3635637.3663048.

[28] J. Makowsky, Why horn formulas matter in computer science: Initial structures and generic examples, Journal of Computer and System Sciences 34 (1987) 266–292. doi:https://doi.org/10.1016/0022-0000(87)90027-4.

[29] B. N. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: Combining logic programs with description logic, in: G. Hencsey, et al. (Eds.), Proceedings of the Twelfth International World Wide Web Conference (WWW), ACM, 2003, pp. 48–57. doi:10.1145/775152.775160.

[30] K. Samuel, et al., Translating OWL and semantic web rules into Prolog: Moving toward description logic programs, Theory and Practice of Logic Programming 8 (2008) 301–322. doi:10.1017/

S1471068407003249.

[31] M. H. van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733–742. doi:10.1145/321978.321991.

[32] A. Martelli, U. Montanari, An efficient unification algorithm, ACM Trans. Program. Lang. Syst. 4 (1982) 258–282. doi:10.1145/357162.357169.

[33] Feigenbaum, Williams, Clark, Torres, SPARQL 1.1 Protocol, W3C Recommendation 21 March 2013, 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/.

[34] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al., SWRL: A semantic web rule language combining OWL and RuleML, Technical Report 79, 2004.

[35] Á. F. Moreira, R. Vieira, R. H. Bordini, J. F. Hübner, Agent-oriented programming with underlying ontological reasoning, in: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies (DALT) Third International Workshop, volume 3904 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 155–170. doi:10.1007/11691792_10.

[36] F. Baader, I. Horrocks, U. Sattler, Chapter 3 Description Logics, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, volume 3 of *Foundations of Artificial Intelligence*, Elsevier, 2008, pp. 135–179. doi:10.1016/S1574-6526(07)03003-9.

[37] V. Mascardi, D. Ancona, M. Barbieri, R. H. Bordini, A. Ricci, CooL-AgentSpeak: Endowing AgentSpeak-DL agents with plan exchange and ontology services, Web Intelligence and Agent Systems 12 (2014) 83–107. doi:10.3233/WIA-140287.

[38] D. Ancona, V. Mascardi, Coo-BDI: Extending the BDI model with cooperativity, in: J. A. Leite, A. Omicini, L. Sterling, P. Torroni (Eds.), Declarative Agent Languages and Technologies (DALT) First International Workshop, volume 2990 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 109–134. doi:10.1007/978-3-540-25932-9_7.

[39] T. G. Halaç, E. E. Ekinci, O. Dikenelli, Description logic based BDI implementation for goal-directed semantic agents, in: O. Boissier, et al. (Eds.), Proceedings of the 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), IEEE Computer Society, 2011, pp. 62–65. doi:10.1109/WI-IAT.2011.192.

[40] T. Klapiscak, R. H. Bordini, JASDL: A practical programming approach combining agent and semantic web technologies, in: M. Baldoni, T. C. Son, M. B. van Riemsdijk, M. Winikoff (Eds.), Declarative Agent Languages and Technologies (DALT) 6th International Workshop, volume 5397 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 91–110. doi:10.1007/978-3-540-93920-7_7.

[41] V. Charpenay, A. Zimmermann, M. Lefrançois, O. Boissier, Hypermedea: A framework for Web (of Things) agents, in: F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, L. Médini (Eds.), Companion of The Web Conference, ACM, 2022, pp. 176–179. doi:10.1145/3487553.3524243.

[42] E. O'Neill, K. Beaumont, N. V. Bermeo, R. W. Collier, Building management using the semantic web and hypermedia agents, in: T. Käfer, A. Harth, A. Ciortea, V. Charpenay (Eds.), Proceedings of the All the Agents Challenge (ATAC), volume 3111 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 32–37. URL: https://ceur-ws.org/Vol-3111/short5.pdf.

[43] A. Ciortea, O. Boissier, A. Ricci, Engineering World-Wide multi-agent systems with hypermedia, in: D. Weyns, V. Mascardi, A. Ricci (Eds.), Engineering Multi-Agent Systems (EMAS) 6th International Workshop, volume 11375 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 285–301. doi:10.1007/978-3-030-25693-7_15.

[44] A. Ricci, M. Piunti, M. Viroli, A. Omicini, Environment programming in CArtAgO, in: R. H. Bordini, M. Dastani, J. Dix, A. E. F. Seghrouchni (Eds.), Multi-Agent Programming, Languages, Tools and Applications, Springer, 2009, pp. 259–288. doi:10.1007/978-0-387-89299-3_8.

[45] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with JaCaMo, Science of Computer Programming 78 (2013) 747–761. doi:10.1016/j.scico.2011.10.004.

[46] R. W. Collier, S. E. Russell, D. Lillis, Reflecting on agent programming with AgentSpeak(L), in: Q. Chen, P. Torroni, S. Villata, J. Y. Hsu, A. Omicini (Eds.), Principles and Practice of Multi-Agent Systems (PRIMA) 18th International Conference, volume 9387 of *Lecture Notes in Computer Science*,

Springer, 2015, pp. 351–366. doi:10.1007/978-3-319-25524-8_22.
[47] F. Meneguzzi, L. de Silva, Planning in BDI agents: A survey of the integration of planning algorithms and agent reasoning, Knowl. Eng. Rev. 30 (2015) 1–44. doi:10.1017/S0269888913000337.
[48] D. Vachtsevanou, A. Ciortea, S. Mayer, J. Lemée, Signifiers as a first-class abstraction in hypermedia multi-agent systems, in: N. Agmon, B. An, A. Ricci, W. Yeoh (Eds.), Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM, 2023, pp. 1200–1208. doi:10.5555/3545946.3598763.
[49] D. Vachtsevanou, J. Lemee, R. Rot, S. Mayer, A. Ciortea, G. Ramanathan, HyperBrain: Human-inspired hypermedia guidance using a large language model, in: Proceedings of the 34th ACM Conference on Hypertext and Social Media, HT '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3603163.3609077.
[50] S. Schmid, M. Freund, A. Harth, Adaptive planning on the web: Using LLMs and affordances for web agents, in: S. Tiwari, B. Villazón-Terrazas, F. Ortiz-Rodríguez, S. Sahri (Eds.), Knowledge Graphs and Semantic Web - 6th International Conference (KGSWC), volume 15459 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 93–108. doi:10.1007/978-3-031-81221-7_7.
[51] P. Novák, J. Dix, Modular BDI architecture, in: H. Nakashima, M. P. Wellman, G. Weiss, P. Stone (Eds.), 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM, 2006, pp. 1009–1015. doi:10.1145/1160633.1160814.
[52] M. Baiardi, S. Burattini, G. Ciatto, D. Pianini, Blending BDI agents with object-oriented and functional programming with JaKtA, SN Comput. Sci. 5 (2024) 1003. doi:10.1007/S42979-024-03244-Y.